35장 스프레드 문법

ES6에서 도입된 스프레드 문법(spread syntax) ... 는 하나로 뭉쳐 있는 여러 값들의 집합을 펼쳐서 개별적인 값들의 목록으로 만든다.

스프레드 문법을 사용할 수 있는 대상은 순회할 수 있는 이터러블에 한정된다.

```
const list = ... [1, 2, 3]; // SyntaxError
```

❗ 스프레드 문법은 값을 생성하는 연산자가 아니다. 따라서 스프레드 문법의 결과는 변수에 할당할 수 없다.

35.1 함수 호출문의 인수 목록에서 사용하는 경우

```
const arr = [1, 2, 3];

Math.max(...arr); // 3
Math.max(arr); // NaN
```

💡 [Rest 파라미터 vs 스프레드 문법]
Rest 파라미터는 함수에 전달된 인수들의 목록을 배열로 전달받기 위해 매개변수 이름 앞에 ...을 붙이는 것
스프레드 문법은 여러 개의 값이 하나로 뭉쳐 있는 배열과 같은 이터러블을 펼쳐서 개별적인 값들의 목록을 만드는 것
따라서 Rest 파라미터와 스프레드 문법은 서로 반대의 개념

35.2 배열 리터럴 내부에서 사용하는 경우

35.2.1 concat

```
// ES5
var arr = [1, 2].concat([3, 4]);
```

```
console.log(arr); // [1, 2, 3, 4]

// ES6
const arr = [...[1, 2], ...[3, 4]];
console.log(arr); // [1, 2, 3, 4]
```

스프레드 문법을 사용하면 별도의 메서드를 사용하지 않고 배열 리터럴만으로 2개의 배열을 1개의 배열로 결합할 수 있다.

35.2.2 splice

```
const arr1 = [1, 4];
const arr2 = [2, 3];
// arr1.splice(1, 0, arr2); // ES5 [1, [2, 3], 4]
arr1.splice(1, 0, ...arr2); // ES6
console.log(arr1); // [1, 2, 3, 4]
```

35.2.3 배열 복사

```
// ES6
const origin = [1, 2];
const copy = [...origin];

console.log(copy); // [1, 2]
console.log(copy === origin); // false
```

원본 배열의 각 요소를 얕은 복사하여 새로운 복사본을 생성한다.

35.2.4 이터러블을 배열로 변환

```
const sum = (...args) => args.reduce((pre, cur) => pre + cu
r, 0);

console.log(sum(1, 2, 3,)); // 6
```

Rest 파라미터 args는 함수에 전달된 인수들의 목록을 배열로 전달받는다.

이터러블이 아닌 유사 배열 객체는 스프레드 문법의 대상이 될 수 없다.

ES6에서 도입된 Array.from 메서드를 사용하여 이터러블이 아닌 유사 배열 객체를 배열로 변경 할 수 있다.

35.3 객체 리터럴 내부에서 사용하는 경우

stage 4 단계에 제안되어 있는 스프레드 프로퍼티를 사용하면 객체 리터럴의 프로퍼티 목록에서도 스프레드 문법을 사용할 수 있다.

스프레드 프로퍼티는 Object.assign 메서드를 대체할 수 있는 간편한 문법이다.

```
// 객체 병합. 프로퍼티가 중복되는 경우 뒤에 위치한 프로퍼티가 우선권을 갖는다.
const merged = { ... { x:1, y:2 }, ... { y:10, z:3} };
console.log(merged); // { x:1, y:10, z:3 }

// 특정 프로퍼티 변경
const changed = { ... {x: 1, y: 2 }, y: 100 };
console.log(changed); // { x:1, y:100 }

// 프로퍼티 추가
const added = { ... {x:1, y:2}, z: 0};
console.log(added) // { x:1, y:2, z:0 }
```